Unsupervised Salient Object Detection with Spectral Cluster Voting

Gyungin Shin[1] Samuel Albanie[2] Weidi Xie[1,3]
[1] Visual Geometry Group, University of Oxford, UK
[2] Department of Engineering, University of Cambridge, UK
[3] Shanghai Jiao Tong University, China
gyungin@robots.ox.ac.uk

In this supplementary material, we first describe the algorithm for spectral clustering (Sec. A). Then, we briefly review the overall structures of the convolution- and transformer-based image encoders and how we extract dense features to which the spectral clustering is applied from each type of encoder (Sec. B). The evaluation metrics are described in Sec. C, and the full results for the comparison between k-means and spectral clustering with different cluster numbers, *i.e.*, $k = \{2, 3, 4\}$ on the three main saliency benchmarks are shown in Sec. D. Lastly, we describe typical failure cases of our model in Sec. E.

A. Normalised spectral clustering algorithm

Here, we describe the normalised spectral clustering algorithm used to generate pseudo-masks for our model in Alg. 1.

ALGORITHM 1 Normalised spectral clustering [8, 11]

Input: An adjacency matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, the number of clusters k to be constructed.

1: Compute the degree matrix $\mathbf{D}$ with $\mathbf{W}$.
2: Compute the unnormalised Laplacian $\mathbf{L}$ using $\mathbf{W}$ and $\mathbf{D}$ using Eqn. 3.
3: Compute the first k generalised eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of the generalised eigen problem $\mathbf{Lu} = \lambda \mathbf{Du}$.
4: Let $\mathbf{U} \in \mathbb{R}^{N \times k}$ be the matrix containing the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ as the columns.
5: For $i = 1, \ldots, N$, let $\mathbf{y}_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of $\mathbf{U}$.
6: Cluster the vectors $\{\mathbf{y}_i \mid i = 1, ..., N\} \in \mathbb{R}^{N \times k}$ with k-means into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$.

Output: Clusters $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

Note that the adjacency matrix $\mathbf{W}$ is computed using Eqn. 2 of the main paper, given the dense features from a visual encoder described next.

B. Visual encoder

Our approach utilises image representations learned by either convolution-based or transformer-based architectures to which spectral clustering will be applied. Here, we first briefly review how these feature representations are computed with each model.

B.1. Convolution-based visual encoder

Convolutional neural networks (CNNs) for image representations, denoted by Φ_{CNN}, consist of a series of 2D convolutional layers and non-linear activation functions which operate on an image in a sliding window fashion. Specifically, given an image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, the CNNs outputs dense feature maps $\mathbf{F}_{\text{CNN}} \in \mathbb{R}^{h \times w \times D}$ where $h{=}\frac{H}{s}$ and $w{=}\frac{W}{s}$ with s denoting the total stride of the network and D denotes the dimensionality of the features. That is,

$$\mathbf{F}_{\text{CNN}} = \Phi_{\text{CNN}}(\mathbf{I}) \in \mathbb{R}^{h \times w \times D} \tag{1}$$

where the parameters for the CNNs are omitted for simplicity.

B.2. Transformer-based visual encoder

In the recent literature, Transformer-based architectures have shown tremendous success in the computer vision community, including ViT [4], DeiT [13], T2T-ViT [18], and BEiT[1]. Generally speaking, these architectures consist of three components, namely, tokeniser (Φ_{TK}), linear projection (Φ_{LP}), and transformer encoder (Φ_{TE}):

$$\mathbf{F}_{\text{Transformer}} = \Phi_{\text{TE}} \circ \Phi_{\text{LP}} \circ \Phi_{\text{TK}}(\mathbf{I}) \in \mathbb{R}^{h \times w \times D} \tag{2}$$

where $\mathbf{F}_{\text{Transformer}}$ denotes the dense features from a transformer-based encoder.

Tokeniser. Given an image as input, *i.e.* $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, the image is first divided by a tokeniser into non-overlapping patches of a fixed size $P \times P$, ending up N patches per frame, *i.e.* $N = \frac{HW}{P^2}$:

$$\Phi_{\text{TK}}(\mathbf{I}) = \{\mathbf{x}_i \mid \mathbf{x}_i = \Phi_{\text{TK}}(\mathbf{I})_i \in \mathbb{R}^{3P^2},\ i = 1, ..., N\} \tag{3}$$

Model	Arch.	Cluster.	DUT-OMRON [17]				DUTS-TE [15]				ECSSD [12]			
			k=2	k=3	k=4	avg.	k=2	k=3	k=4	avg.	k=2	k=3	k=4	avg.
Fully-supervised features														
ResNet [6]	ResNet50	k-means	.311	.346	.355	**.337**	.345	.358	.360	**.354**	.461	.445	.425	**.444**
ResNet [6]	ResNet50	spectral	.258	.326	.346	.310	.297	.341	.343	.327	.424	.454	.432	.437
ViT [4]	ViT-S/16	k-means	.335	.406	.440	**.394**	.349	.423	.460	**.411**	.505	.560	.562	.542
ViT [4]	ViT-S/16	spectral	.268	.392	.481	.380	.260	.428	.511	.400	.402	.613	.637	**.551**
Self-supervised features														
MoCov2 [5]	ResNet50	k-means	.334	.387	.403	.375	.401	.423	.422	.415	.507	.511	.481	.500
MoCov2 [5]	ResNet50	spectral	.311	.399	.453	**.387**	.403	.464	.496	**.454**	.602	.642	.638	**.627**
SwAV [2]	ResNet50	k-means	.356	.412	.429	.399	.415	.456	.462	.444	.548	.552	.526	.542
SwAV [2]	ResNet50	spectral	.346	.407	.450	**.401**	.412	.473	.488	**.458**	.594	.606	.569	**.590**
DINO [3]	ViT-S/8	k-means	.299	.381	.427	.369	.299	.385	.447	.377	.497	.566	.591	.551
DINO [3]	ViT-S/8	spectral	.315	.417	.463	**.398**	.311	.435	.486	**.411**	.527	.616	.618	**.587**
DINO [3]	ViT-S/16	k-means	.314	.391	.426	.377	.325	.407	.444	.392	.507	.557	.560	.541
DINO [3]	ViT-S/16	spectral	.310	.413	.459	**.394**	.324	.445	.483	**.417**	.528	.609.	.596	**.577**

Table 1: Comparison between k-means algorithm and spectral clustering with three different cluster sizes k on the three benchmarks. IoU between a ground-truth mask and the closest prediction among k predicted masks is considered. On the fourth column of each benchmark, we report the average of the results from the different k. The higher average scores of k-means and spectral clusterings within the same model are in bold.

where $\mathbf{x}_i$ denotes the ith patch.

Linear projection. Once tokenised, each patch from the image is fed through a linear layer Φ_{LP} and projected into a vector (a.k.a. token):

$$\mathbf{z}_i = \Phi_{\text{LP}}(\mathbf{x}_i) + \text{PE}_i \in \mathbb{R}^D$$

where $\mathbf{x}_i \in \mathbb{R}^{3P^2}$ refers to the ith patch, and its corresponding learnable positional embeddings $\text{PE}_i \in \mathbb{R}^D$ are added to the patch token $\Phi_{\text{LP}}(\mathbf{x}_i) \in \mathbb{R}^D$. Then, the N augmented patch tokens are concatenated altogether with a *class token* `[CLS]` $\in \mathbb{R}^D$, producing the final input form of $\mathbb{R}^{(N+1)\times D}$ to a sequence of transformer layers, described in the following.

Transformer encoder. A transformer encoder is composed of multiple transformer layers, each of which is subdivided into a self-attention layer and multi-layer perceptrons (MLPs). The self-attention layer contains three learnable linear layers, each of which takes the input tokens and outputs either key K, value V, or query Q of the same dimensionality as the input tokens, i.e., $\mathbb{R}^{(N+1)\times D}$.[1] Then, the self-attention layer outputs $\texttt{softmax}(\frac{QK^T}{\sqrt{D}}, \texttt{dim=}-1)V \in \mathbb{R}^{(N+1)\times D}$.[2] The outputs of the attention layer are fed to the following MLPs, which are composed of two linear layers with a non-linear activation between them and output tokens with their shape preserved.

[1]In practice, we use a single linear layer which maps the D dimension of the input tokens to $3 \times D$ and equally splits them into Q, K, and V.

[2]Here, we consider the single head case for simplicity. For more details, please refer to the original paper [14].

Note that, as all transformer layers constituting the transformer encoder share the identical architecture, the final outputs from the ViT have the same shape as the input tokens, i.e., $\mathbb{R}^{(N+1)\times D}$. For image classification task, only the `[CLS]` $\in \mathbb{R}^D$ is taken from the outputs and fed to a linear classifier. In our work, however, we consider the patch tokens $\mathbf{F}_{\text{Transformer}} \in \mathbb{R}^{N\times D}$ which are reshaped to $\mathbb{R}^{h\times w\times D}$ where h and w equal $\frac{H}{P} \times \frac{W}{P}$. It is worth noting that, the patch size P plays the role of total stride as in the CNNs.

C. Descriptions of evaluation metrics

In the following, we describe the metrics used for evaluation:

- F_β [10] is the harmonic mean of precision and recall between a ground-truth $G \in \{0,1\}^{H\times W}$ and a binarised mask $M \in \{0,1\}^{H\times W}$:

$$F_\beta = \frac{(1+\beta^2)\text{Precision} \times \text{Recall}}{\beta^2\text{Precision} + \text{Recall}}, \tag{4}$$

  where β^2 denotes a weight of precision.[3] Following previous work [16, 9, 19, 7], we set β^2 to 0.3, putting more weight on precision. We use F_β to compute the maximal-F_β, described next.

- maximal-F_β (maxF_β) is a maximum score of F_β among multiple masks binarised with different thresh-

[3]Precision$= \frac{tp}{tp+fp}$ and Recall $= \frac{tp}{tp+fn}$ where tp, fp, and fn represent true-positive, false-positive, and false-negative, respectively.

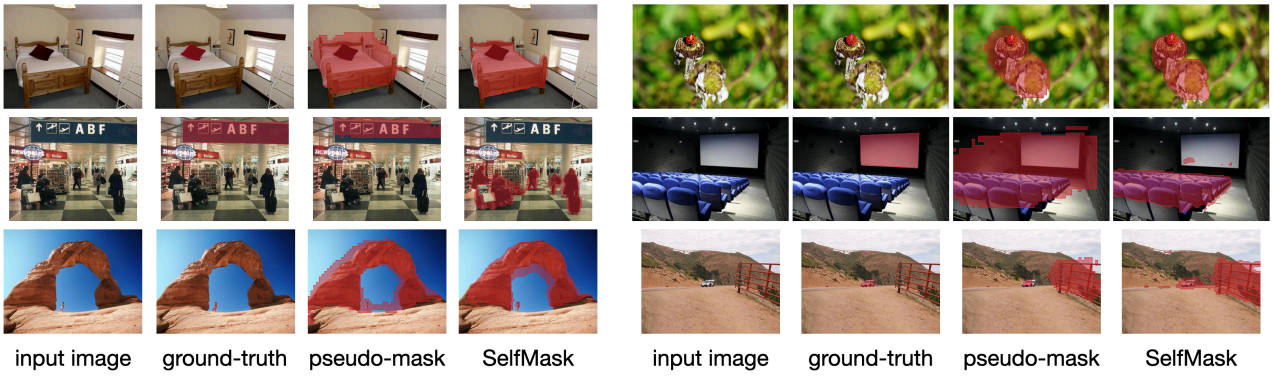

Figure 1: Sample visualisations for typical prediction failures from our model on the DUT-OMRON [17] and DUTS-TE [15] benchmarks. From left to right, input image, ground-truth mask, a pseudo-mask, and a predicted mask by our model are shown. The respective salient regions are highlighted in red. Best viewed in colour. Please zoom in for details.

olds. Specifically, given a non-binarised mask prediction with its value between [0, 255], it computes F_β from 255 binarised masks, each of which is thresholded by an integer among $\{0, ..., 254\}$ and takes the maximum F_β value for the result.

- Intersection-over-union (IoU) is the size of overlapped foreground regions between a ground-truth G and a binarised mask prediction M divided by the total size of foreground regions from G and M.

- Accuracy (Acc) is a metric that measures pixel-wise accuracy based on a ground-truth mask G and a binarised mask prediction M:

$$\text{Acc} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \delta_{G_{ij}, M_{ij}} \tag{5}$$

where δ denotes the Kroneker-delta.

D. Comparison between k-means and spectral clustering

In Sec. 4.3 of the main paper, we show the performance of k-means and spectral clustering applied to different architectures (i.e., ResNet50 and ViT-S/$\{8, 16\}$) and features (i.e., fully- and self-superivsed features) averaged over k=$\{2, 3, 4\}$ on the three saliency datasets. Here, we show the full results for each k in Tab. 1. For the description, please refer to Sec. 4.3 of the main paper.

E. Visualisation of failure cases

In Fig. 1, we visualise some failure predictions from our model on the DUT-OMRON [17] and DUTS-TE [15] datasets.

We notice there are two typical failure cases. First, when a salient object is of small scale, the model tends to under-segment it and prefers the large salient object. For instance, as shown by the top left example in Fig 1, the whole bed is segmented, rather than the pillow; Second, when there are more than one salient region in the image, our model may only segment one of them. For example, as shown by the middle right example in Fig 1, both screen and seats can be thought of as a salient region while the model only highlights only the latter. We conjecture that these cases are caused by a bias of the dataset (*i.e.*, DUTS-TR [15]) on which the model is trained. That is, the training images likely to contain large salient regions composed of either an object or objects sharing a semantic meaning, thus discouraging the model from predicting a small salient region or more than one object with different semantics even if all the objects can be regarded salient.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2022.

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[7] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *CVPR*, 2019.

[8] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2004.

[9] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Finding an unsupervised image segmenter in each of your deep generative models. *arXiv:2105.08127*, 2021.

[10] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 2012.

[11] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

[12] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *TPAMI*, 2015.

[13] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers &; distillation through attention. In *ICML*, 2021.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[15] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017.

[16] Wenguan Wang, Qiuxia Lai, Huazhu Fu, Jianbing Shen, Haibin Ling, and Ruigang Yang. Salient object detection in the deep learning era: An in-depth survey. *TPAMI*, 2021.

[17] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013.

[18] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

[19] Jia-Xing Zhao, Jiang-Jiang Liu, Deng-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. Egnet: Edge guidance network for salient object detection. In *ICCV*, 2019.